what this text covers . . .

Problem Analysis and Decision Tables 1

This first section tells why you should first analyze a computer problem. The section also discusses decision tables and illustrates how the programmer can make effective use of such tables.

Flow Charting 9

This section discusses both the system flow chart and the program flow chart. You'll learn about flow-chart symbols, the main line of a flow chart, housekeeping, looping, and various flow-charting techniques and conventions.

Multiple Card Layout Form 41

The uses of a multiple card layout form are described in this section. You will learn how to analyze a source document when planning card fields on a multiple card layout form. Classification of information and how to divide information between two cards are also discussed.

Printer Spacing Chart 47

This final section presents a thorough discussion of the printer spacing chart.

26396C/8408 Printed in South Africa

problem analysis and decision tables

Analyzing a Computer Problem

1. The first step in preparing an application for a computer is to define the problem thoroughly. In addition, program planning requires a complete analysis of all parts of the problem. This includes analysis of the input information, the logical and practical procedures that will be needed to solve the problem, and the form of the final output. This analysis must take place before coding of the program can begin. Otherwise, time may be wasted in preparing a program that will be subject to major change before completion.

Most computer programs are too complex and contain too many instructions to be understood as a whole. For that reason, individual segments of the program should be planned separately. No programmer can keep in mind all of the possible answers to all of the decisions in his program. A typical payroll program, for example, may require answers to questions about payroll deductions, overtime, employee status, quarterly and yearly tax reports, and labor distribution.

There are a number of documentary tools available that a programmer can use to identify the various requirements, as well as the input and output for coding a program. Some of these tools for analysis are more important than others; the most important of all is the flow chart. In addition, a pro-

grammer may undertake to prepare decision tables before drawing up flow charts. After flow charting, the programmer will want to know all he can about the format of both the input data and the output data.

In this unit we will first tell you what a decision table is and how to use it. Second, we will discuss flow charting in detail, including both systems and program flow charts. Last, we will discuss the purpose of both the multiple card layout form and the printer spacing chart and the methods of using them.

Decision Tables

2. The purpose of a decision table is to provide information concerning problems and solutions in a concise format that is easy to read and understand. A decision table defines all conditions (the prerequisites for an action) and separates them from all actions. Furthermore, it relates given conditions to the appropriate actions.

When a programmer is writing a program for a computer, he must tell the computer to do one thing and not to do another, because the result of his program is based on the existence of certain conditions. An example would be to order more of an item if the last sale of that item has reduced the on-hand balance to a predetermined reorder point.

A decision table developed to handle numerous situations like the preceding will contain a list of both the possible conditions and the possible results related to the problem. These are called condition statements and action statements. A condition statement describes a condition that must be present (inventory below reorder point) before a certain action can take place (buy or reorder to build inventory to a predetermined level). An action statement contains a description of what must be done if something else is present. You can distinguish between condition statements and action statements by thinking of the word "if" as pertaining to a condition and the word "then" as indicating the need to take action. For example, *if* it rains today *then* the picnic is canceled. Another example would be: *if* the program is correctly written, *then* the computer will correctly solve the problem.

In the preceding examples the conditions that must be present are "if it rains" and "if the program is written correctly." The two key words "if" and "then" may not always be stated,

but a programmer can use them to help himself distinguish between condition and action statements.

3. It is, of course, possible for the result of an action to be based on two conditions, both of which must be met before an action can or need be taken. If, for instance, you satisfactorily complete several tests and pass the final exam, you will receive a certificate showing that you successfully completed some required work. In an *and situation* an action does not take place until more than one condition is satisfied. (Example: If instruction A is correct *and if* instruction B also is correct, then do instruction C.) If neither condition of an "and" statement is satisfied, an alternative action will have to be taken. Usually this occurs when the word "otherwise" could be used as part of the result. If, for instance, you fill out a time card and sent it to payroll, you will be paid; *otherwise* you will not be paid.

When a decision in a computer program can be made by meeting any one of a number of conditions, we refer to this as an "or" situation. For instance, if a new item does not have a catalog number or a unit price, do not include it in ending inventory. In other words, an *or situation* will exist when any of several conditions can produce an action.

4. Decision tables can be used independently of, or to complement, flow charts. You will have to decide this based on the complexity of the computer program that you will be coding. You will find, however, as you become a more experienced programmer, that it is better to resolve as many questions as possible about a problem before you attempt any coding or computer testing. Many programmers develop a lazy habit of letting the computer tell them, by program testing, what they missed. As a result of taking this approach, a large amount of time is lost in completing a program. Even with the most careful problem analysis, program testing on a computer is time-consuming.

Obviously it is unwise to rush into a job of program coding without the proper programming tools, which include problem statements, decision tables, and flow charts. Decision tables are related to the use of decision symbols in flow charting and branching instructions in coding. You will see this when these topics are discussed.

5. Decision tables are divided into four sections, as shown in Fig. 1. The section labeled *1* will contain the *condition statements;* the section labeled *2* will contain the *result state-*

ments; the section labeled *3* will contain a pattern of *yes* and *no* answers to the conditions; and the section labeled *4* will use a capital *X* next to an action statement to remind a programmer to code for a certain action in his program.

Condition statements are located above the double horizontal lines and to the left of the double vertical lines. Action statements are located below the double horizontal lines and to the left of the double vertical lines.

6. An example of the use of a decision table containing two condition statements is shown in Fig. 2. Note that this is an example of the use of the word *or*. If either one of two conditions is present here, we want the program to take a specified action. If neither condition is present, we want the program to take a different specified action.

In Fig. 2 there are two conditions that must be tested: 1) If amount is negative and 2) if amount is zero. The Y and N codes are used to indicate a *yes* condition or a *no* condition. Two possible results are indicated for this problem: 1) Do *not* calculate commission and 2) calculate commission. An X next to either of these results will indicate when to take that action. A second example of the use of a decision table is shown in Fig. 3.

We will review the concept of decision tables before we go on to flow charting. For this purpose we have prepared a more complicated problem statement, and also the decision table needed to understand the many decisions that must be made relative to the problem. This is the kind of problem that definitely requires a decision table. Without the table as a guide, the programmer could quite easily omit many of the possible alternative decisions when he coded his program.

In Fig. 4 is an example of an airline reservation system problem. Fig. 5 is the decision table used to analyze the problem.

When constructing a decision table you may avoid a certain amount of confusion by not filling in all of the Y and N boxes for all conditons. If a yes answer to one condition results in an action, all the other condition boxes in the vertical column can be left blank. An example of this is shown in Fig. 6. If you follow this procedure, there will be less chance for you to forget a condition statement when you are building your table.

1	3
2	4

CONDITIONS	YES OR NO ANSWERS TO THE CONDITIONS
ACTIONS	WHAT ACTION SHOULD BE TAKEN

FIGURE 1. FORMAT, DECISION TABLE

IF AMOUNT IS NEGATIVE OR ZERO, DO NOT CALCULATE COMMISSION; OTHERWISE CALCULATE COMMISSION.

AMOUNT IS NEGATIVE	Y	Y	N	N
AMOUNT IS ZERO	Y	N	Y	N
DO NOT CALCULATE COMMISSION	X	X	X	
CALCULATE COMMISSION				X

FIGURE 2. EXAMPLE, DECISION TABLE

IF EMPLOYEE WORKS OVER 40 HOURS WEEKLY AND IS NON-MANAGEMENT PAY HIM OVERTIME; IF EMPLOYEE IS MANAGEMENT AND WORKS OVER 40 HOURS, GIVE HIM ADDITIONAL VACATION TIME. IF EMPLOYEE DID NOT WORK OVER 40 HOURS PROCESS HIS PAYROLL IN REGULAR MANNER.

IS EMPLOYEE MANAGEMENT	Y	N	Y	N	
DID EMPLOYEE WORK OVER 40 HOURS	Y	N	N	Y	
PAY OVERTIME				X	
GIVE ADDITIONAL VACATION TIME	X				
PROCESS IN REGULAR MANNER		X	X		

FIGURE 3. EXAMPLE, DECISION TABLE

1. PROBLEM STATEMENT: If the request is for a first-class reservation and a first-class is available, issue a first-class ticket and subtract one from the first-class seats available. However, if a first-class seat is not available but tourist class is and the passenger will accept tourist class, issue a tourist class ticket and subtract one from tourist availability. If passenger will not accept tourist class, or tourist is not available, place on first-class waiting list. If the original request was for tourist class and tourist class is available, issue a tourist class ticket and subtract one from tourist availability. If tourist class is not available, but first-class is and passenger will accept first-class, issue a first class ticket and subtract one from first class availability. Otherwise, place on tourist waiting list.

FIGURE 4. PROBLEM STATEMENT, DECISION TABLE

Solution is: CONDITIONS: 1st Class Request	1 Y	2 Y	3 Y	4 Y	5	6	7	8
Tourist Request					Y	Y	Y	Y
1st Class Open	Y	N	N	N		Y	N	
Tourist Open		Y	N		Y	N	N	N
Alternate Class Acceptable		Y	Y	N		Y	Y	N
ACTIONS: Issue 1st Class Ticket	1 X	2	3	4	5	6 X	7	8
Issue Tourist Ticket		X			X			
Subtract 1 from 1st Class Avail.	X					X		
Subtract 1 from Tourist Avail.		X			X			
Place on Tourist Wait List			X				X	X
Place on First Class Wait List			X	X			X	

FIGURE 5. DECISION TABLE, SOLUTION

IF AMOUNT IS NEGATIVE OR ZERO, DO NOT CALCULATE COMMISSION; OTHERWISE CALCULATE COMMISSION.

AMOUNT IS NEGATIVE	Y	Y	N	N
AMOUNT IS ZERO			Y	N
DO NOT CALCULATE COMMISSION	X	X	X	
CALCULATE COMMISSION				X

FIGURE 6. EXAMPLE, DECISION TABLE

Check Your Learning

"Check Your Learning" is a test of your ability to retain the information presented in this textbook. It is a simple yet highly effective test consisting of incomplete statements for which you must supply the missing word or words to make the statement true. Study each statement carefully, and then answer it as best you can. ALL ANSWERS CAN BE FOUND AT THE END OF THE TEXT. DO NOT SEND YOUR ANSWERS OR SOLUTIONS TO US. Check each of your answers with the answer given. If your answer does not agree, review the text material to find out why.

1. Decision tables are made up of _____ statements and _____ statements.
2. Decision tables are divided into _____ sections.

flow charting

Purpose of Flow Charts

7. Flow charting is the most important step in solving data processing problems. Development of any flow chart requires the work of a person who is capable of analyzing a problem in terms of the sequence in which events take place. Flow charting is the stepping-off point to coding and program testing on the computer.

The program flow chart should eliminate all assumptions on the part of the programmer prior to coding. Developing a flow chart will naturally bring to light further questions about the problem. The programmer-analyst must resolve these questions while they are fresh in his mind and before he goes ahead with his flow charting. Programmers may or may not construct decision tables, but flow charting is always a necessity prior to coding.

8. Flow charts are used to show both the present and the proposed flow of data for a typical business application, the operations involved, and the sequence in which the operations occur. Flow charts use specific symbols to represent the sequence in which a series of operations is to be performed. It is important that you learn to identify these symbols so that you will use them correctly. Because these symbols are used for communication between different groups of data processing personnel, they must be standardized. Standardization is essential, especially if the person who analyzes a

problem and designs a flow chart is not the same person who will write the computer program. It often happens that the programmer who develops a flow chart later resigns from the company, and his work is then turned over to some other programmer. Or after a program has run for a number of years it may have to be modified by some programmer other than the person who wrote the original program. The use of standardized symbols and a clear knowledge of what they mean will help a new programmer understand the flow charts of other programmers.

Fig. 7 shows the percentages of overall time used to accomplish the different functions involved in resolving a typical computer problem. Once a complete flow chart has been drawn for an application, the job is usually better than one-third completed. According to the chart, problem analysis requires 25% of the total time and flow charting 10%.

Analyzing and defining a programming problem, which includes the drawing of a good flow chart, is considered to be a more important job than that of translating flow-charted procedures into computer instructions.

Types of Flow Charts

9. The purpose for which a flow chart is intended determines how it will be developed. Generally speaking, there are two types or levels of flow charts—the system flow chart and the program flow chart. The type we shall discuss first is the top level or *system flow chart.*

System Flow Chart

10. The system flow chart which represents the broader approach of the two types of charts, is an excellent tool for showing the total picture of a business system; and it is useful either when studying an existing business system or when designing a new one. The system flow chart shows how documents pass through various work stations. This chart can also be used to illustrate manual, mechanical, or computer operations, or any mixture of these operations. Emphasis is placed on the flow of information throughout a system and on the general sequence of operations rather than on how each operation is performed. Fig. 8 shows a simplified version of a system flow chart; it stresses the flow of information, with little regard for the details of the operation involved.

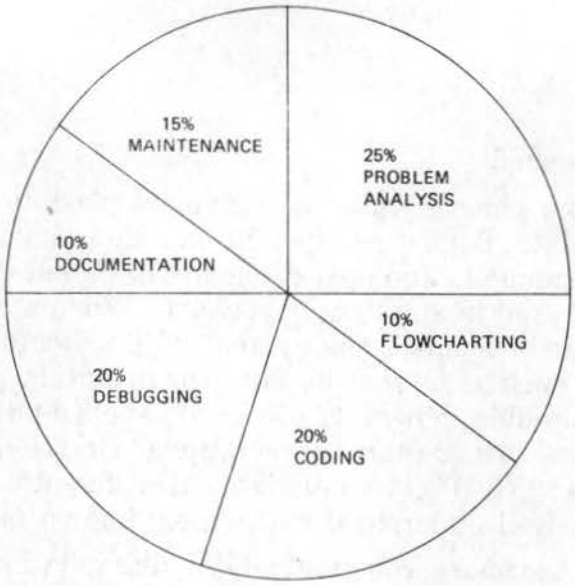


FIGURE 7. AVERAGE TIME SPENT ON A PROBLEM BY A PROGRAMMER

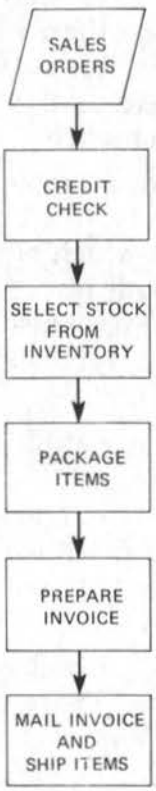


FIGURE 8. SYSTEM FLOW CHART

Flow Chart Symbols

11. A flow chart symbol is a symbol used to represent operations, data, flow, or equipment on a flow chart. Symbols depicting documents and operations are designed to be meaningful when used in a system flow chart. When a comment is placed within a symbol, the symbol will indicate a specific application, such as sales order entry or inventory processing. Wherever possible, however, use clear, spelled-out language on flow charts rather than abbreviations. On the other hand, symbols like "YTD" (year-to-date) are acceptable because they have a well-understood and widely known meaning.

Some of the more commonly used abbreviations are the following:

$=$	Equal to
$\neq$	Not equal to
$>$	Greater than
$<$	Less than
$\geqslant$	Greater than *or* Equal to
$\leqslant$	Less than *or* Equal to
:	Compare

Fig. 9 provides another example of a system flow chart, this time in much greater detail. This chart is an application of inventory updating, in which old inventory item balances are adjusted by receipts and issues that bring about current item inventory balances. Fig. 9 also shows how comments are used to explain what the symbols mean in the particular application.

The system flow chart is established prior to the preparation of the program flow chart. It is not as formalized as a program flow chart, and it is more flexible. This difference will be discussed later. A greater choice of symbols is available in developing system flow charts and more freedom is allowed in their use.

12. Examples of standardized system flow charting symbols are shown in Fig. 10. There are three basic flow chart symbols: the symbol for input/output; the symbol for processing; and the symbol for flow direction. As shown in Fig. 11, a system flow chart could be drawn using only the three basic symbols. But in addition to the three basic symbols illustrated in Fig. 11, there are fourteen other symbols used to describe more specifically the equipment, machine operations, and manual operations involved in handling any flow of data. As you noted, all seventeen symbols are shown in

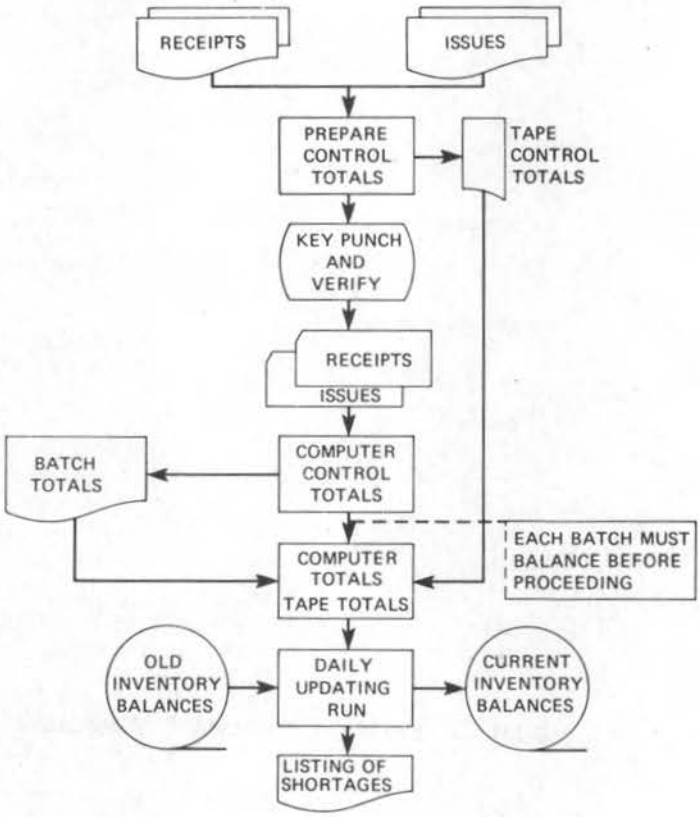


FIGURE 9. INVENTORY UPDATE

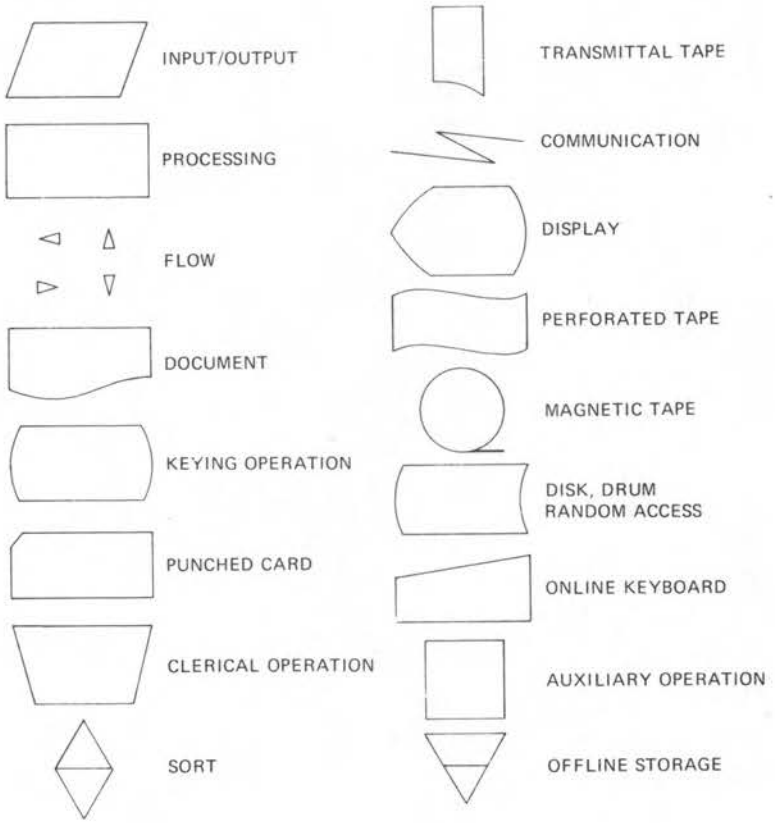


FIGURE 10. SYSTEM FLOW-CHART SYMBOLS

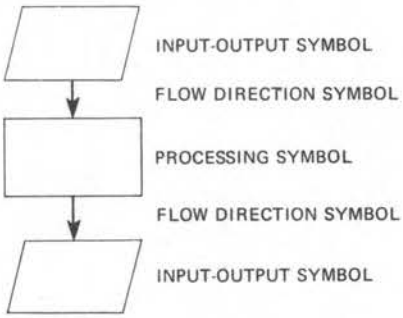
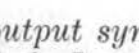
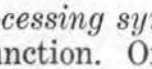
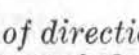


FIGURE 11. THREE BASIC SYMBOLS OF SYSTEM FLOW CHART

Fig. 10. These symbols are the ones prescribed for information processing by the International Organization for Standardization. Each of these seventeen symbols is now explained.

The *input/output symbol*  can represent medium or data of any kind. It can also be used in place of a specific symbol, such as a punched card or magnetic tape.

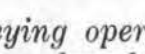
The *processing symbol*  represents a major data processing function. Often a system flow chart will include a number of these symbols, and each of them may indicate a separate computer program that will have to be written out. This symbol can also be used to indicate a manual operation, such as filing, or a mechanical action, such as operating an adding machine.

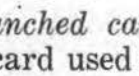
The *flow of direction symbol*  represents the direction of processing or information flow. Most flow charts are read from top to bottom and from left to right.

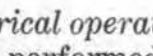
Arrowheads must be used on lines which oppose the general direction of data flow; if desirable, they may be used for all lines. A broken line is generally used to indicate that the output of the processing in one period becomes input to the processing in the next period.

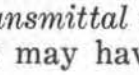
The remaining symbols are used to make a system flow chart more specific and easier to understand. These symbols stand for different input, output, and processing methods.

The *document symbol*  represents paper documents and reports of all varieties, including source documents.

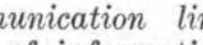
The *keying operation symbol*  represents an operation using a key-driven device, such as an adding machine, keypunch, or typewriter.

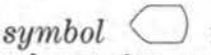
The *punched card symbol*  represents any type of punched card used as either input or output.

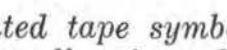
The *clerical operation symbol*  represents any manual operation performed on documents before or after they are processed.

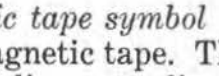
The *transmittal tape symbol*  represents any control tape that may have been prepared either manually or by machine.

The *sort symbol*  represents an operation where data is put in sequence.

The *communication link symbol*  represents the transmission of information from one location to another over communication lines. In Fig. 12 is an example of the use of this symbol.

The *display symbol*  represents information that can be shown on display units, such as a cathode-ray tube, as part of computer output.

The *perforated tape symbol*  represents perforated tape when this medium is used as input or output of a system.

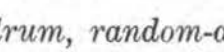
The *magnetic tape symbol*  is used to represent data recorded on magnetic tape. The symbol is made up of a circle with a straight line extending from the bottom of the symbol and to the right. Fig. 13 shows a section of a system flow chart which illustrates the use of the magnetic tape symbol. There is also a broken line to indicate that an output tape produced as a result of today's operation will be the input tape for processing when this job is run again.

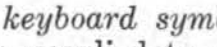
The part of the flow chart shown in Fig. 13 shows the following operations taking place in the processing unit:

1. An hourly rate from a daily payroll master tape is multiplied by the number of hours worked on each time card. This computation is performed on the basis of a match of employee number from the master tape to the time card.

2. Total cost and hours are added to accumulated amounts on the master payroll tape, and a new tape is written with current totals.

3. The new total hours and amounts are printed out on a "Daily Performance Report."

The *disk, drum, random-access file symbol*  is used to indicate the storage of master or transaction files on auxiliary storage devices.

The *online keyboard symbol*  is used to show how information is supplied to a computer from a key-driven device that is connected to the computer. This type of data

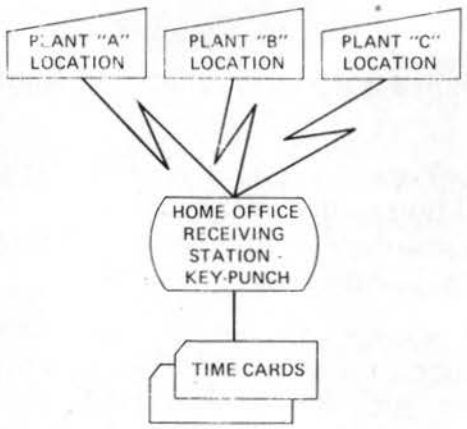


FIGURE 12. TRANSMISSION OF DATA OVER COMMUNICATION LINES

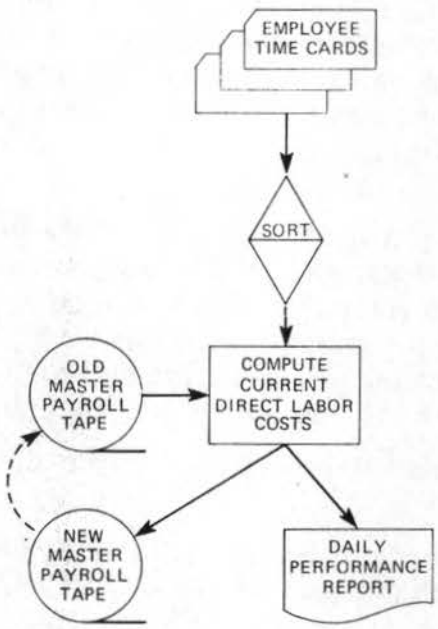


FIGURE 13. DIRECT LABOR COST UPDATING

is usually supplied over communication lines from terminal stations.

The *auxiliary operation symbol*  represents an extra machine operation which supplements the main processing function. An example would be separating carbon sheets from multiform documents.

The *off-line storage symbol*  represents information that has been obtained from a file and is ready for processing, or information, such as punched cards, that has been processed and is now being returned to file storage.

Program Flow Charts

13. The second type of flow chart is called a program flow chart. It evolves from a system flow chart and includes a greater amount of detail, such as the program logic used for coding and the specific processing sequences.

A program flow chart is a graphic representation of a computer program. Like the systems flow chart, it uses symbols to represent machine instructions or groups of instructions. Computer programmers have found this diagrammatic aid helpful in coding a program without becoming lost in details.

Specifically, a program flow chart is a diagram of operations and decisions, and of the sequence in which they are performed by a computer. In developing a program for the computer, the programmer uses flow charting to organize his facts; to outline problem logic and solutions; and to deal systematically with the entire programming application.

The main considerations in program flow chart preparation are:

1. A standardized technique for flow charting should be used so that others will be able to follow it.
2. The program should have a clearly defined beginning and ending.
3. The work flow on the chart should be all in the same direction, as this reduces confusion.
4. No flow line should stop or be unconnected at any point.
5. Program flow charts should be prepared in such a way that another programmer could program from it if

necessary, even though this situation may not actually arise.

6. Reference notes should be used to describe complex logic or unique computer program concepts.

A detailed program flow chart, when completed, will show the following:

1. A graphic picture of the problem solution.
2. A graphic presentation of program logic to be used for coding, desk checking, and debugging while testing.
3. Verification that all possible condition and action relationships have been considered.
4. Program documentation for future reference.

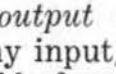
Basic Program Flow-Chart Symbols

14. In the initial stage of writing a program, the flow chart serves as an aid to the programmer in developing an orderly, logical program. Generally, the programmer begins by using those symbols which represent the major functions of any proposed computer program. The programmer develops the overall logic of a program by developing the major functions in more detail, until one symbol on the flow chart represents one instruction to be coded.

The three basic flow chart symbols used in system flow charts are also the three basic symbols used in program flow charts.

a. The *flow direction*  *symbol.* This symbol indicates the direction of the processing flow. The general flow is top to bottom, and from left to right. In order to show a clear, logical flow of operations, it is suggested that you avoid crossing-over flow lines.

b. The *processing*  *symbol.* This symbol is defined as the symbol used to represent general processing functions not represented by other symbols.

c. The *input/output*  *symbol.* This symbol is used to indicate any input/output function. Making information available for processing is an input function, and recording processed information is classified as an output function.

A program flow chart may be drawn with only three symbols, as shown in Fig. 14.

Special Flow-Chart Symbols

15. Special processing symbols are also used to make program flow charts more meaningful. These symbols may be used in place of the basic processing symbols in program flow charting. Descriptions of these special symbols follow.

a) The *decision* ◇ *symbol* is used to indicate a point in a program where a branch to one of two or more alternate paths is possible. (A branch instruction causes the computer to switch from one point in a program to another point, thereby controlling the sequence in which operations are performed.) Here the manner in which a choice is to be made should be clearly indicated. Such decisions may be based on a comparison, or on the test of an indicator, such as an end-of-file indicator (a code which signals that the last record of a file has been read).

After the last card in a deck of cards has been read and processed, the programmer may wish the computer to print out accumulated totals. The signal that will tell the computer to do this is the last-record indicator. In order to remind the person who is coding the program to include a "totals" operation, it will be shown on the flow chart in the form of a decision symbol. The symbol will indicate two alternative paths in the program, one to be taken at "last-card time" and the other to be taken for all other cards.

Fig. 15 is an example of the use of a decision symbol.

b) The *program modification* ⬡ *symbol* is used to indicate an instruction or a group of instructions. An example would be address modification—the process of changing the address part of a machine instruction by means of coded instructions.

c) The *terminal* ⬭ *symbol* is used to indicate any point at which a program begins or ends. It is also used to indicate a computer halt when data errors are detected by the program, such as cards out of sequence.

d) The *connector* ○ *symbol* represents an entry from, or an exit to, another symbol on the same page. Fig. 16 illustrates the use of a connector symbol. Thus a connector symbol may be used to represent a break in a single flow line, or

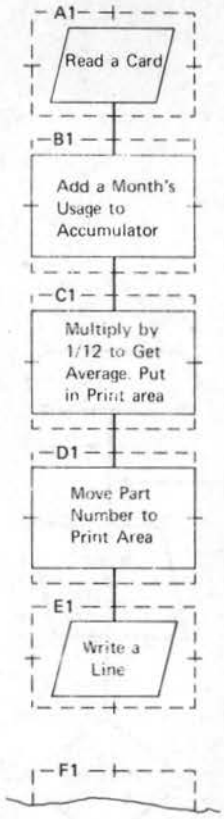


FIGURE 14. PROGRAM FLOW CHART DRAWN WITH THREE SYMBOLS

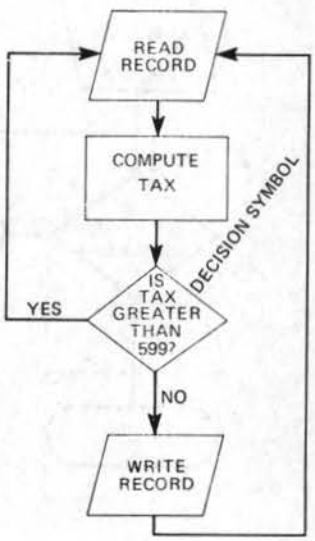


FIGURE 15. PART OF PROGRAM FLOW CHART TO ILLUSTRATE USE OF DECISION SYMBOL

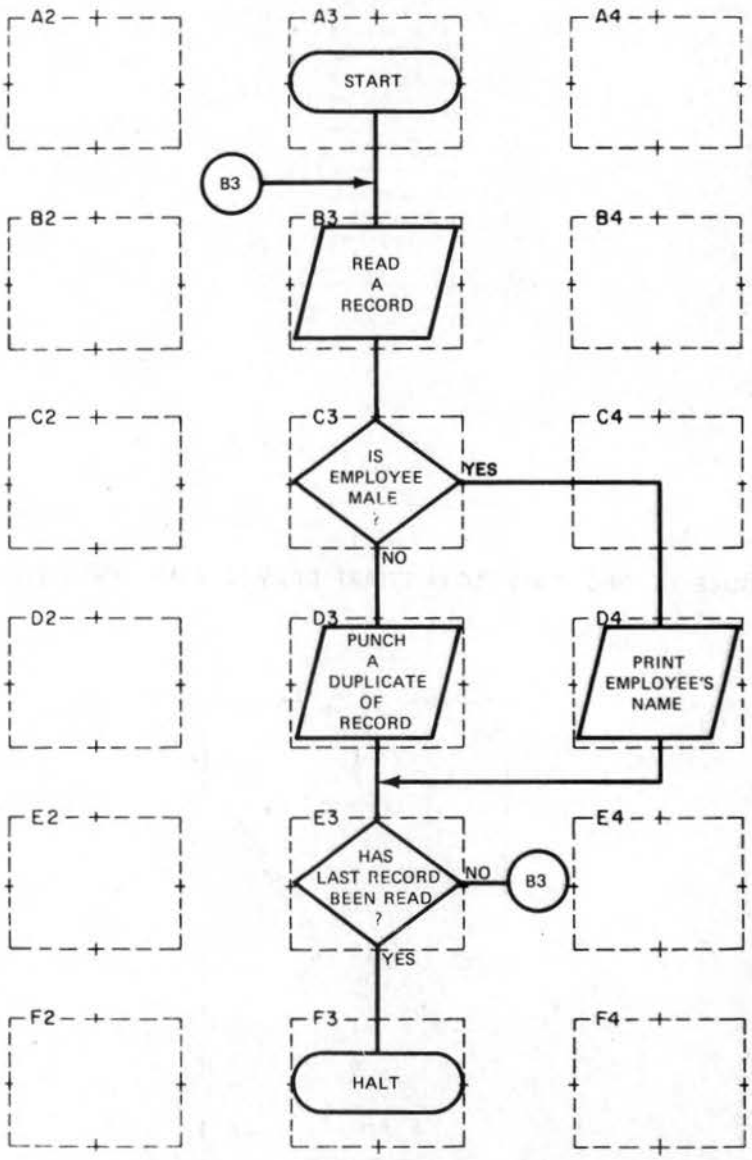


FIGURE 16. USE OF CONNECTOR SYMBOLS

to show continuation into another area. On a flow chart the flow line is a line which represents a path between flow-chart symbols. A set of two connector symbols can also be used to indicate a continued flow. Connector identification should always be placed within the symbol.

e) The *off-page connector* ▽ *symbol* is used to indicate exit from, or entry to, a page. When using this symbol, place it so that its point indicates the direction of the flow. Connector identification should always be placed within the symbol.

f) The *predefined process* ▯ *symbol* is used to represent a group of operations that are not detailed on the flow chart. An example of the use of this symbol is to represent small programs, called subroutines, which have been written by other programmers (IBM) and which we may want to include in our own program.

Main Line of a Flow Chart

16. In Fig. 17 the sequence of steps reads from top to bottom—from A to E. This idea flow, which is typical of all programs, is referred to as the *main line.* Main line is the sequence normally followed in representing the major job to be accomplished, and it serves to show the correct direction in which to read the flow chart.

The steps following each of the letters B, C, and D indicate the presence of logical *decision points*—points at which circumstances can bring about a deviation from the main line. In a detailed flow chart of an actual program, such *decision points* may add up to hundreds of steps.

Because the main line of a program represents the most important sequence in solving that program, the good programmer will pay close attention to the way he develops the main line of his flow chart. His goal, of course, will be to minimize the computer time required to process the main line, thereby assuring a workable, efficient program.

Charting

17. Any wording on the program flow chart takes the form of spelled-out English words rather than machine language. Titles used with the symbols should be short and descriptive; and care should be taken that the titles are not confusing. Condense explanations inside the symbols so that

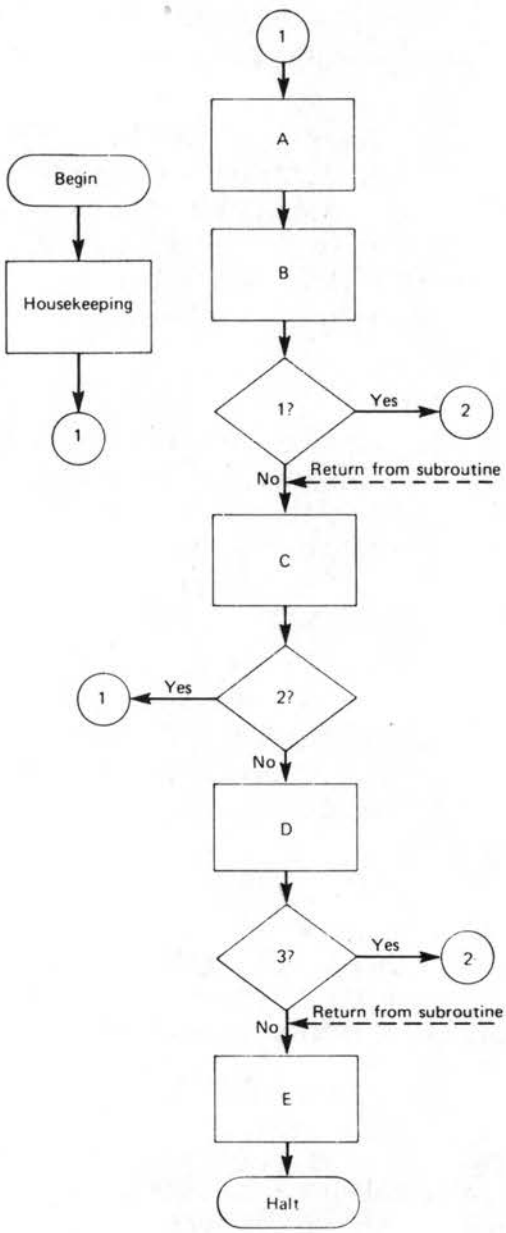


FIGURE 17. MAIN LINE OF A FLOW CHART

they fit without crowding. Use abbreviations cautiously; above all, do not use ambiguous abbreviations. Remember that the processing steps start at the top of the page and progress downward to the lower right-hand corner of the page.

Multiple exits from a decision symbol can be shown by having several lines branching from the symbol to other symbols, or by having a single line branching from the decision symbol and that line branching into any number of other single lines. Fig. 18 is an example of multiple line exits.

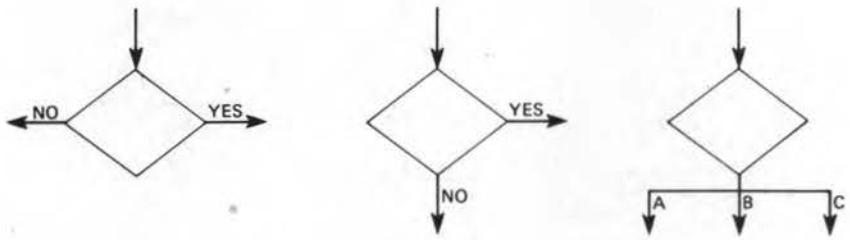


FIGURE 18. MULTIPLE DECISION SYMBOL EXITS

So that you may better understand the use of program flow charting symbols, we will show you first a fairly uncomplicated problem statement and its flow-charted solution. Fig. 19 presents the problem to be flow charted; Fig. 20 presents the solution in flow-chart form.

PROBLEM STATEMENT

JOB

Inventory Cost Analysis

DESIRED RESULTS

1. If Part Number begins with numeral "3", list card on printed report.

2. If not:

   calculate Average Quantity Per Month = total of Quantity in Warehouse for four months divided by four;

   calculate Average Cost Per Month = Average Quantity Per Month multiplied by Cost of Part;

   assign Analysis Code "A" to cards where Average Cost Per Month is higher than Cost Level, and Analysis Code "B" to other cards;

   punch four identical cards for each part, containing all data on input card plus Average Quantity Per Month, Average Cost Per Month, and Analysis Code.

   when last card has been read and processed, notify the operator, and stop

FIGURE 19. PROBLEM STATEMENT

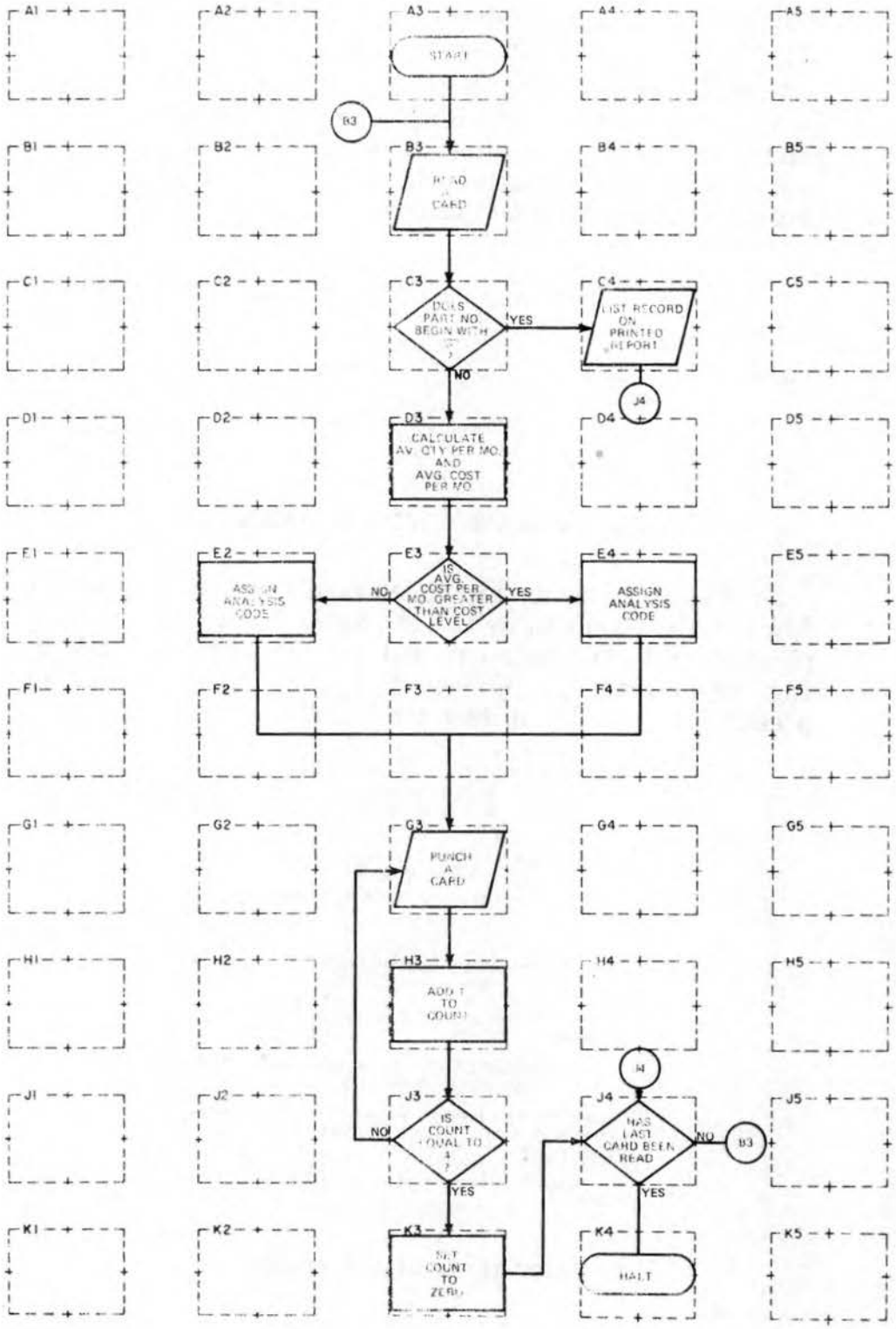


FIGURE 20. SOLUTION IN FLOW-CHART FORM

18. Flow charts are usually much more involved than the one in Fig. 20. The flow chart shown in Fig. 21 was developed from an analysis of an "Accounts Receivable Updating" problem statement.

The flow-charting techniques used in Fig. 21 include the following:

a) Connector symbols are used in place of flow lines which would cross other lines.

b) Multiple exits from the decision symbol are shown by several lines running from the symbol.

c) Arrowheads are used when the flow directions are not from top to bottom or from left to right.

The Accounts Receivable Updating program is used for posting transactions such as returns, cash receipts, and invoices to customers' accounts. The program shown in Fig. 21 indicates only major operations.

The mechanics of flow charting must be understood before the operation of any program is considered. For this reason, a full explanation of the program chart follows.

The reference block at the top of Fig. 21 is used to identify the chart, page, program, and other necessary data. Alphabetic notations are used for chart identification and page identification, thereby permitting the use of many combinations of alphabetic numbers in two positions.

The chart itself has five vertical rows, with ten blocks in each row. Each of these blocks has an identification or reference number in the upper left-hand corner. This number is used in connector and off-page connector symbols to indicate a change in the sequence of operations.

Examination of the input-output used in this program shows two primary input files: the accounts receivable master file and the transaction file. The accounts receivable (A/R) master file (a file which contains relatively permanent information) is comparable to an A/R ledger. This file would be stored on some form of external storage medium, such as magnetic tape. The transaction file (a file which contains data to be processed in conjunction with a master file) consists of adjustments, cash receipts, and billings. This data is usually punched in cards, but it can also be on tape. The master file and the transaction file are placed on separate input devices so that one file can be read (to trans-

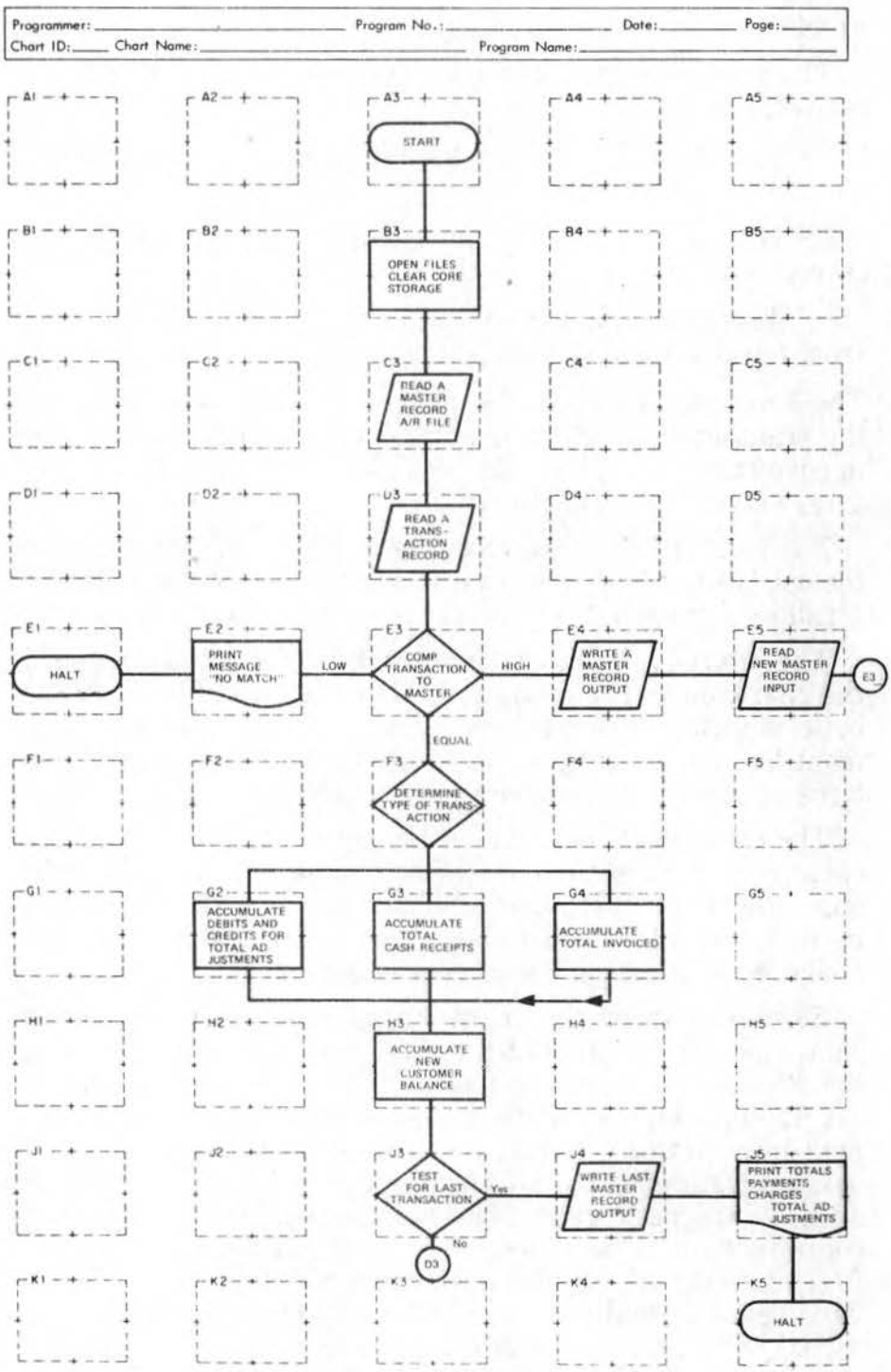


FIGURE 21. FLOW CHART OF AN ACCOUNTS RECEIVABLE UPDATING PROGRAM

late source data into machine language) without interfering with the other file. Both the master file and the transaction file must be in ascending sequence by customer number.

An updated A/R master file, being the old master file updated with the transactions by the program, is the output—that is, data which has been processed. This output file is on the same type of storage device as the input (data to be processed) A/R master file. However, a different physical device is used so that writing (which is the process of recording data on a storage device or a data medium) on this file does not interfere with any other input-output operation. The printer prints any error messages—that is, indications that an error has been detected—as indicated in block E2. Totals of adjustments made to customer balances will be printed after all transactions are processed as shown in block J5.

Examination of the program flow chart in Fig. 21 shows a terminal symbol at block A3 on page AA. (A terminal symbol is a point in a system or communication network at which data can enter or leave.) The processing symbol at block B3 (open files, clear storage area) represents a housekeeping procedure. The first A/R master record is read at C3. The first transaction record is read at D3. At E3 a comparison is made on customer number between a transaction record and a master record. Since both files must be in ascending sequence, a transaction record that is low will indicate "no-match" with a master record. Depending on what the procedure is for a non-match, you can halt the computer, as indicated at E1, or you may want to continue running by branching back to D3 and reading the next transaction record.

If the transaction record is high in relation to the master record, this will indicate that a new master record should be compared with the present transaction record. However, before we read and process a new master record, we must write out the present master record on the updated file in order to retain it on the current file. Next, we will want to go to our master record input file for a new item to compare against the present transaction record. We see at E5 the reading in of a new master record and then the branch back to E3 as shown by the connector. The reason we did not branch back to C3 to read a master record is that this path would have taken us through D3, and at this stage we do not want to read a transaction record.

If a transaction record and a master record are equal, we will want to update the customer's account balance; that is, we want to process the transaction record. To update the customer's account, we have to subtract payments and add invoice charges. Whether a particular transaction record will be added or subtracted from an account balance is usually determined by a code punched in the card, and so we have to test this card column to determine the type of transaction.

After the new balance has been computed, the next step will be to determine if the transaction record we just finished processing was the last item. There are many different methods of determining this fact, varying with different computers. At this time all you have to know is that there may be "total time" operations to be performed. If the transaction record is the last item, you will want to update the last customer balance on the output master record and to print the report totals for payments, receipts, and total adjustments.

If the item just processed was not the last transaction record, the program should branch back to D3, as shown by the connector at the bottom of the page. At D3 the program will read in a new transaction record whose customer number may be higher than, or equal to, the previous transaction record.

If the new transaction record has a customer number higher than the master record, this is the cue that tells the computer that it is time to write out an updated master record and to read in a new master record for processing, as shown at E5 and connector E3. After the processing of one transaction record and branching back to D3, if the next transaction record has the same customer number, this means that we have more than one transaction for the same customer. This second transaction will be processed like the first one. It should be noted that new customer balances are computed at H3 but written out on tape at E4 on the basis of a new transaction record having a higher customer number than the master record in storage. The only other place where a master record is written out is at J4, which accounts for the last item before the totals.

Housekeeping

19. The first instructions of nearly every program are intended to perform functions of housekeeping in preparation

for processing. These instructions are used to set storage to an initial condition, clear registers, establish constant values or set program switches.

Information once stored in computer memory will remain there for use until it is replaced by new information. Information from the running of a prior program could be included by mistake in a subsequent program if the same areas of core storage were used in the second program. Consequently it is important to reset all core areas used in a program to blanks or zeros at the beginning of a program.

In addition, housekeeping instructions may perform system checks by testing to determine whether all input/output units required by the program are ready for operation. The information pertinent to the proper operation of the system may be called to the operator's attention by programmed instructions.

It is a good idea, when drawing program flow charts, to note the fact that certain housekeeping routines should be included when the program is being coded. Usually these operations take place before the reading of a record, and for that reason any information on housekeeping should be noted alongside the input/output symbol.

Looping

20. Certain calculations involve repeating a number of instructions over and over again. The programmer can avoid wasted storage space by instructing the control unit to "branch" back (repeat the process), rather than write a complete set of instructions for each record read. This repetitive operation is called looping.

Fig. 22 is a flow chart which calls for an input card to be read, its contents to be moved to the punch area, and the input card data to be punched into a blank card. In a 500-card input deck, the cards would be processed via the loop until the 500th card had been processed. Without this looping feature the programmer would have to write out a 1500-step program, duplicating the first three steps for each input card in the same deck. Every time the program flow charted in Fig. 22 is looped back to process another card, it executes a pass. In

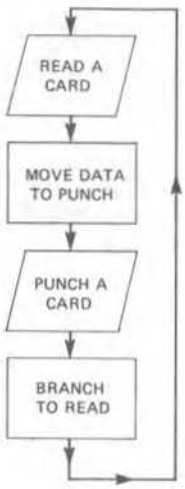


FIGURE 22. FLOW-CHART FORM OF LOOPING

this instance it will take 500 passes to process the entire deck of 500 cards fed in for processing.

The necessary elements for a loop consist of the initial data, the instructions required to perform the operation, some sort of counter (a device used to represent the number of occurrences of an event), and a storage location in which intermediate results are temporarily stored. The loop begins when a counter is given a predetermined starting value, usually a 1 or a 0, as part of the "housekeeping" function. Then the operation is performed and the result of the operation is stored. Next, the counter is tested against a predetermined value. If a comparison does not indicate the end of the operation, a fixed amount is added to the counter, usually a +1, and the entire process is repeated. When the counter reaches its end value, the loop is broken and the program either ends or branches to another part for other processing. This concept will be explained in more detail in Part 4.

Flow-Charting Techniques and Conventions

21. One of the most important uses of a program flow chart is to provide the programmer with a means of visualizing the sequence in which arithmetic and logical operations should occur, and also the relationship of one portion of a program to another portion.

In the program development stage, a flow chart may serve as a means of experimenting with various approaches in developing the logic of a program. Here the programmer first develops the overall logic of the program. This is accomplished by detecting the input and output functions, the steps necessary to identify and to select the required records, and the necessary decision functions.

Once the overall main-line logic of the program is determined, large segments of the program being worked on are taken from the main-line logic flow chart and described in considerable detail. This must be done if the programmer is to produce flow charts which clearly show all the major decision points in the program. In addition, detailed flow-chart documentation can be used to make certain that the program's procedure satisfies all the possible conditions which can arise after the program is tested and being used in actual production.

When the procedure is established and has been proved to be sound, the program flow chart becomes a guide to coding. The amount of flow-chart detail depends, of course, upon the purpose the flow chart is to serve. Although it is not generally done by programmers, it is a good idea to go back and redraw your flow chart after your program has been properly tested. This will assure you of having an up-to-date flow chart that corresponds to your correct coding. When you change a program that has been running correctly on the computer, do not forget to modify your flow chart.

After the program has been coded, it should be documented so that it can be used to facilitate any future modifications which may be required while testing, installing, and operating the program. Since the flow chart serves as a map of the program listing, it simplifies any problems which arise because of program modification. To be useful in program modification, the symbols used on a flow chart must be related (by labeling) to coded instructions.

The final documentation of a program should include both the overall main-line logic flow chart and the detailed flow charts. Main-line logic flow charts promote understanding of the more detailed flow charts, and they also give you a clearer realization and understanding of the procedures to be coded.

22. Some of the more detailed techniques of program flow charts will now be described.

Identification. Processing steps should start at the top of the page and progress down and to the lower right corner. Each page should be correctly sequenced and should be identified with application name, program name and number, flowchart number, and title of the program.

Cross-referencing. Cross-referencing is used to relate the program flow chart to the source-language programs. (The source language is the original form in which a program is coded prior to processing by the machine.) Cross-referencing is helpful in program debugging, maintenance, and modification.

One way to cross-reference is to relate the instructions on a coding sheet to the symbols on a flow chart by placing the number of the instruction next to the symbol. The cross-reference can be placed above the upper left corner of a symbol, as shown in Fig. 23.

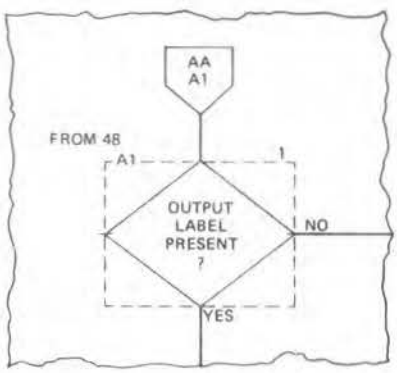


FIGURE 23. CROSS-REFERENCING SYMBOL

Striping. Striping on a symbol (the horizontal line within and across the upper part of a symbol) is used to indicate a complex logical program unit. Generally, the use of striping indicates that a more detailed flow chart of that part of the program is available. Identification of the program unit may be placed above the stripe, while a brief description of its function may be placed below the stripe. Fig. 24 illustrates how the striping technique can be used.

Decision Techniques. Decision techniques may be shown in several ways. One such technique is illustrated in Fig. 25. The decisions made here will determine which action will be taken next by the program.

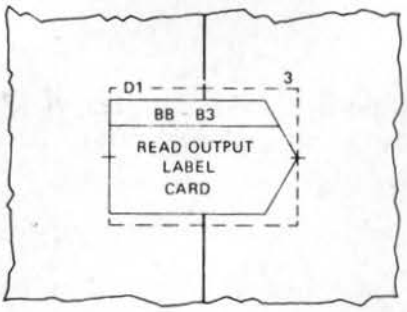
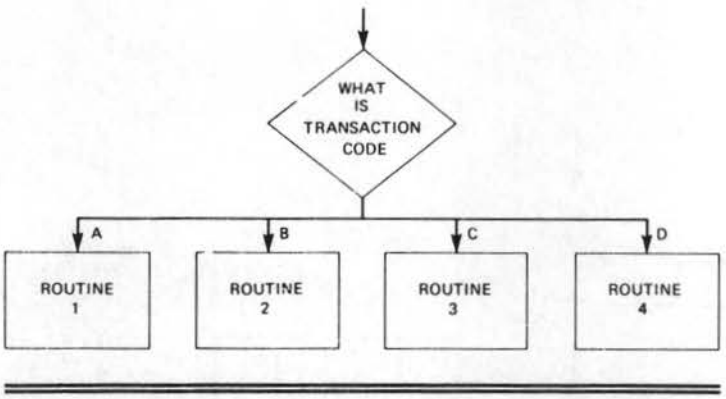


FIGURE 24. EXAMPLE OF STRIPING



CORRESPONDING INSTRUCTIONS IN PROGRAM:

Instruction	Operations
* * * * * * * * * *	
2010	If Trans. Code is A, go to Inst. 2050
2011	If Trans. Code is B, go to Inst. 2065
2012	If Trans. Code is C, go to Inst. 2080
2013	Go to Inst. 2035
* * * * * * * * * *	
2035	(First inst. in Routine 4)
* * * * * * * * * *	
2050	(First inst. in Routine 1)
* * * * * * * * * *	
2065	(First inst. in Routine 2)
* * * * * * * * * *	
2080	(First inst. in Routine 3)
* * * * * * * * * *	

FIGURE 25. EXAMPLE OF A DECISION TECHNIQUE

Fig. 26 shows additional examples of decision techniques. Also included are notations commonly used to indicate program conditions.

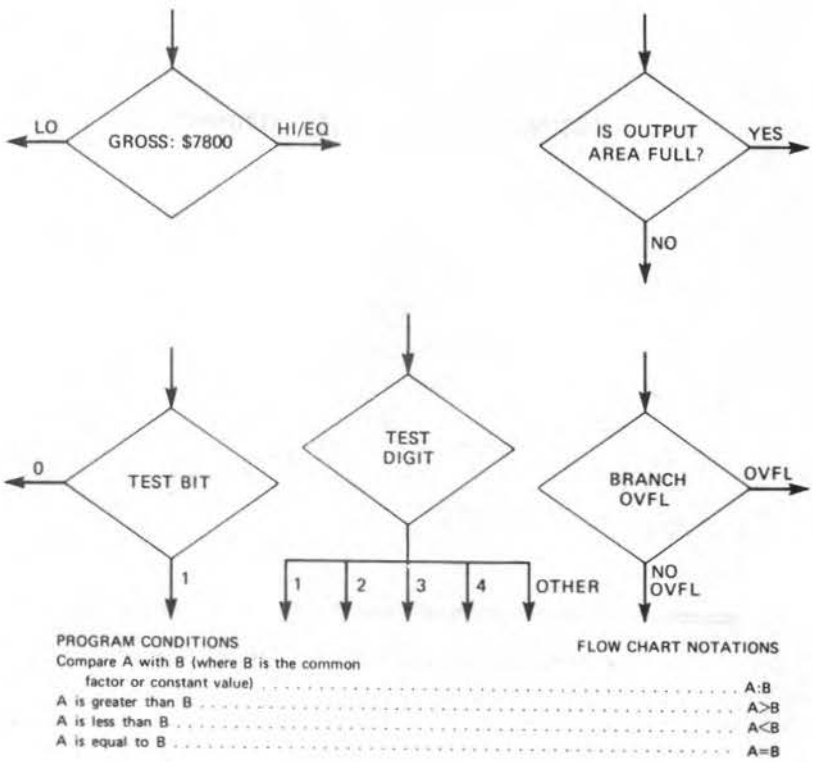
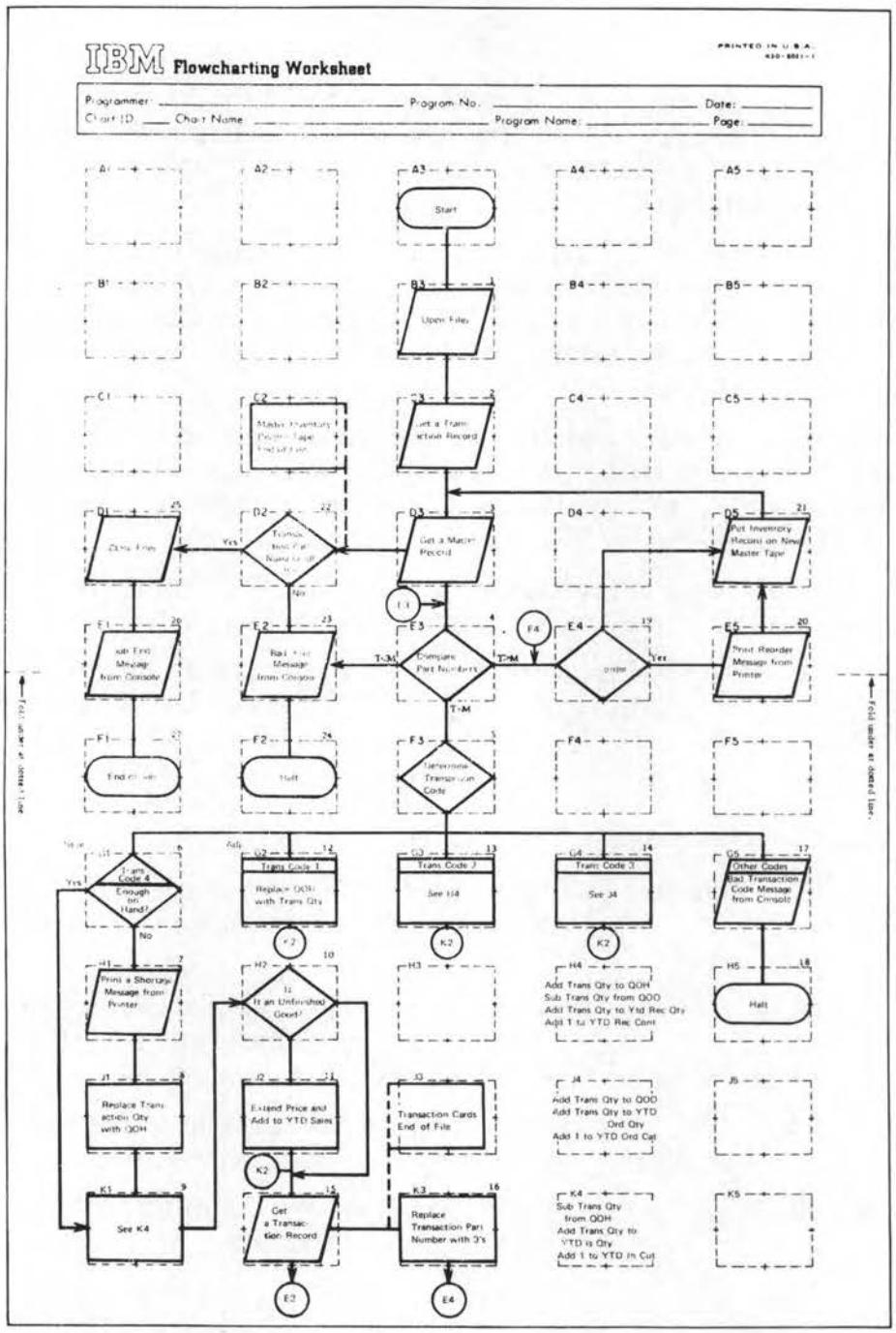


FIGURE 26. EXAMPLES OF DECISION TECHNIQUES

23. Fig. 27 is an example of a detailed program flow chart of an inventory updating run. Presented here are a number of logical data processing concepts which it is not our intention to explain in detail. The purpose of this illustration is to make you better acquainted with the use of symbols and terminology. However, the basic techniques described here apply to all program flow charts.

With the exception of the off-page connector, all the program flow chart symbols are used in Fig. 27. This illustration includes the following:

a. The use of the annotation symbol in two places where clarification to the programmer is required. The dotted lines are connected to the flow line at the entrance to the symbol whose use requires further explanation. (C2-J3)



Courtesy of International Business Machines Corporation

FIGURE 27. A PROGRAM FLOW CHART TO UPDATE INVENTORY RECORDS

b. The development of multiple exits from the decision symbol. The exits are shown both by a single line from the symbol (and its branches to other lines) and by several lines from the symbol. (E3-F3)

c. The use of a stripe on a given processing block. The stripe was placed there to show that a more detailed program flow chart exists. Note that the identification is placed above the line, and that a brief discussion of the processing function is placed below the line. (G3-G4)

d. The use of the connection symbol in place of flow lines, where such flow lines would cross other flow lines. The reference numbers indicate the in-connection or in-location blocks on the worksheet. (G2-G3-G4)

e. The use of arrowheads when the flow of direction is *not* from top to bottom and from left to right. (K1)

f. The numbering of computer program operations at the top and to the right of the symbols so that they can be easily referenced. (E1-F1)

Check Your Learning

3. Flow charts use specific symbols to represent the _____ in which a series of operations is to be performed.
4. In a systems flow chart, emphasis is placed on the _____ _____ _____ throughout a system rather than on how each operation is performed.
5. The _____ symbol is used to represent any operation on collating and sorting equipment.
6. The three basic flow-chart symbols are the input/output, _____, and _____ direction.
7. The _____ operation symbol is used to represent the operation of adding machines and other key-driven devices.
8. The _____ operation symbol is used to represent any extra machine operation which supplements the main processing function.
9. The signal that will tell the computer that the last card in a deck of cards has been read and processed is called the _____-_____ indicator.

10. The idea flow which is typical of all programs is referred to as the _____ _____.
11. In coding computer instructions, the term "housekeeping" refers to the _____ procedure.
12. A repetitive routine in programming is referred to as a _____.

multiple-card layout form

The Input

24. After having clearly defined the problem by constructing a program flow chart, you must define both the input and the output. Actually, it makes little difference whether you define the output or the input first. In some cases, after having defined the output, you can better determine what input is required. Whatever the order, a convenient method of defining input and output is the worksheet layout approach.

If a computer program is going to use punched cards as input data, the programmer must prepare a card layout form before he attempts to code his program. In preparing such a form, and in discussing "input" with the individual who requested the program, he will be checking to see that he has accounted for all the necessary information to produce the required output. It is a good idea to get written confirmation from the person requesting the program after you have discussed the input format with him.

25. The designing of card layout forms will prompt the programmer to get answers to questions like the following:

1. How many different types of cards will I be working with on this problem?
2. Will there be both master and detail cards, and if so, how many detail cards for each master?

3. What will be the volume number of cards each time this job is run on the computer?
4. How many fields of data will there be on each different type of card?
5. Will there be positive and negative quantities in amount cards?
6. What is the maximum size of each card field?
7. Can I use the punched card output of some other program?
8. How long must these cards be kept on file?

26. The next step in developing card layouts is to evaluate the information received in answer to questions, so that you can design the most efficient form of layout. In Figs. 28 to 30 you will examine a program application designed to indicate how a multiple card layout form is used. Fig. 28 is a program request for a "Student Name and Address Listing." Fig. 29 is an explanation in narrative form of what the card input should contain. Notice that there will be a set of three cards for each student. Card column 1 will contain a card code to enable the computer to distinguish between the three types of cards. The student number is the card field (key) common to all three cards. Having a common field in each card of a set is important, simply because it can serve as a means of getting all cards back into proper order if they ever get out of sequence.

Finally, Fig. 30 is a fairly simple illustration of how a set of student cards should look on a multiple card layout form.

Sequence on Source Documents

27. The multiple card layout form can be used as a guide to the keypunch operator in preparing the input punched cards for the program. Therefore the sequence of information as it appears on the source document should be compared with the sequence in which this information will be punched into the card, because the location of the information will definitely affect the speed of the key punch operation. The fields—that is, the assigned areas in a record which are to be marked with information—to be keypunched should be arranged so that information can be found on the source document by reading from left to right and from top to bottom.

PROVIDE A LISTING OF STUDENT NAME AND ADDRESS CARDS

1. PRINT A HEADING ON EACH PAGE.
2. PRINT STUDENT NUMBER AND STUDENT NAME ON THE FIRST LINE' STREET ADDRESS ON THE SECOND LINE, AND CITY, STATE, ZIP CODE ON THE THIRD LINE.
3. DO NOT SEPARATE A GROUP OF CARDS BY PRINTING ONE CARD ON ONE PAGE AND THE OTHER TWO CARDS OF THAT GROUP ON ANOTHER PAGE.
4. DOUBLE SPACE BETWEEN LINES OF A GROUP AND TRIPLE SPACE BETWEEN GROUPS.
5. PRINT A TOTAL NUMBER OF STUDENTS AT THE END OF THE LISTING.

FIGURE 28. PROGRAM REQUEST

STUDENT NAME AND ADDRESS FILE

THE STUDENT NAME AND ADDRESS FILE WILL CONSIST OF THREE CARDS PER STUDENT.

(a) The first card will consist of:

Column 1	"1"
Columns 2-5	Student Number (to be assigned by instructor)
Columns 6-35	First Name, Middle Initial, Last Name

(b) The second card will consist of:

Column 1	"2"
Columns 2-5	Student Number
Columns 6-35	Street Address

(c) The third card will consist of:

Column 1	"3"
Columns 2-5	Student Number
Columns 6-35	City, State, Zip Code

FIGURE 29. CARD INPUT, NARRATIVE FORM

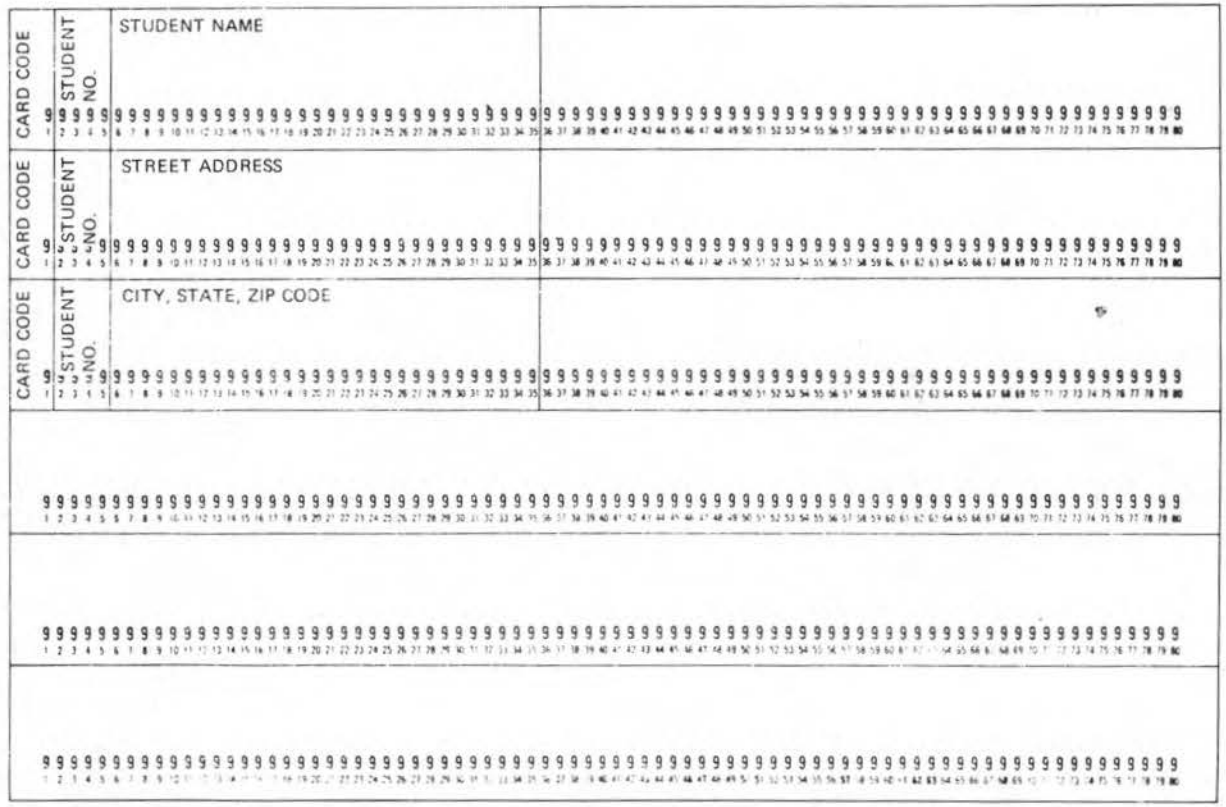


FIGURE 30. CARD INPUT, STUDENT LISTING

Classification of Information

28. Information to be recorded can be classified as being Reference (R), Classification (C), or Quantitative (Q).

Reference information is punched information which can be used to identify the source of the data, such as an invoice or bill of lading.

Classification information is punched information which can be used for further classifying reference information. State, county, and type of customer are examples.

Quantitative information includes punched figures which are to be used in any calculation the results of which will be subsequently punched into a card. Quantity, unit price, and amount, and hours worked, wage rates, tax rates, and net pay are examples of quantitative data. On card layout forms the sequence of information types from left to right is usually reference followed by classification followed by quantitative.

Size of Fields

29. To ensure the completenes of your data, you must provide enough columns for each type of information. The size of certain fields can be easily determined; others require considerable research and calculation. The sizes of the fields required for the various data are usually obtained by determining the largest single number or name to be recorded. A date column, for example, would require six card columns—two columns for the month, two for the day, and two for the year.

The assignment of card columns for fields containing quantitative information usually varies in the number of digits. However, the maximum number of card columns to be reserved cannot always be determined, because occasionally very large amounts may have to be recorded. Rather than reserve card columns that will seldom be used, it may be better to punch extra cards.

Once the total number of digits required for each card field has been decided, the total number of card columns to be used is ascertained and this total placed at the bottom of the worksheet. If the total is less than 80, the figures are acceptable for the card, since the maximum card size is 80 columns. If the total exceeds 80, it must be decided whether to use two cards or to reduce the number of card columns to 80 or fewer.

If two cards are to be used, one of the following methods of dividing information between the cards may be used:

a) The use of master cards and detail cards. Here some of the information, such as name and address, or customer number, may be constant for each source document. This information would appear on the master card. Other detail information, such as quantity purchased, price, and amount, may vary with each source document and so would appear on the detail card. The two cards must, however, contain one duplicate item of information (key) that can be used to bring the cards together. A customer account number would serve to do this.

b) The use of a card containing data for one report and of a second card containing data for a by-product report. Although the information for both reports will come from the same source document, one card will be punched with information relative to one report and the other card will be punched with information relative to another report.

c) The use of cards which are divided into three groups. This method can be used in preparing bills or invoices. Here one group of cards is designated as heading cards; these cards will contain the customer's name and address, the shipping address, and the terms of sale. A second group of cards, miscellaneous data cards, will contain such information as salesman identification, routing of shipment, and bill of lading. The third group of cards, detail commodity cards, will contain a description of the product sold, and the quantity, price, and amount of the sale. This method is similar to that described in *(a)*, involving the use of a master card and a detail card.

Check Your Learning

13. The multiple-card layout form can be used as a guide by the _____ operator in preparing the input punched cards for the program.
14. Information to be recorded can be classified as being _____, _____, or _____.
15. When using both a master card and a detail card to contain information referring to one transaction, both cards must contain one duplicate item of information, known as a _____.

printer spacing chart

The Output

30. Just as a programmer must design the input specifications for card fields and for different types of cards, so he must determine how data will appear on a printed report. The standard form used for designing report formats is called the printer spacing chart. This chart, which is really a work sheet, will be used for reference when the programmer is ready to code that section of his program dealing with the printed output.

Fig. 31 shows one type of printer spacing chart.

You will notice that there are 51 numbers listed on the left-hand side of the form; these are "line numbers." Across the top of the form we have 144 printing position numbers which correspond to the print bars on a printer. The largest printer currently in use has 144 printing positions on a line. You will also notice that there are vertical dotted lines drawn at positions 100, 120, 132, and 144. These are to remind the programmer that he must not exceed a particular point if he is writing a program for a computer that has a printer with only so many print positions.

The section to the left of the line numbers is used to indicate the proper positioning of the paper in the printer. A carriage-control tape that looks like this one is punched and

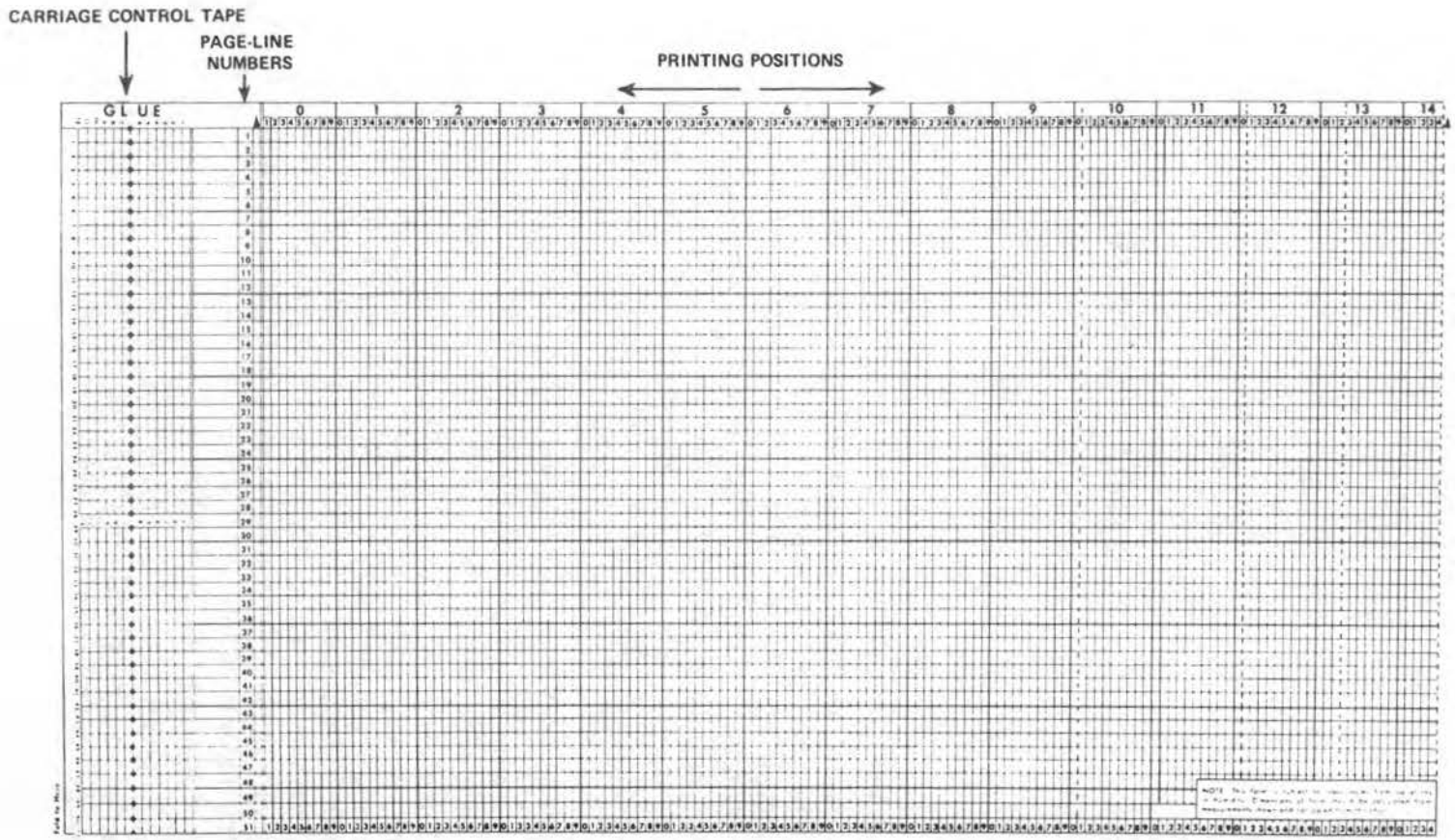


FIGURE 31. PRINTER SPACING CHART

inserted in the printer for each job. At the top of this tape, under the word "glue," are 12 small numbers, known as channels. For each report, marks are made on this tape to indicate at what line number the first line of a report should be printed on each page. The mark for first-line printing is usually made under channel 1 at whatever line number will be the first print line. Another mark is made to indicate where the last printing line on a page should be, usually under channel 12 at a line number that represents the last line of detail print on a page. The other 10 channel numbers at the top of the tape are used to indicate where the skipping or spacing of more than one line will occur on any page of a report.

The programmer or computer operator will physically punch up a carriage-control tape, glue the two ends together in circular fashion, and insert the tape on a wheel in the printer. The size of any tape will be cut to correspond to the exact number of lines on one sheet of paper. A carriage-control tape is shown in Fig. 32.

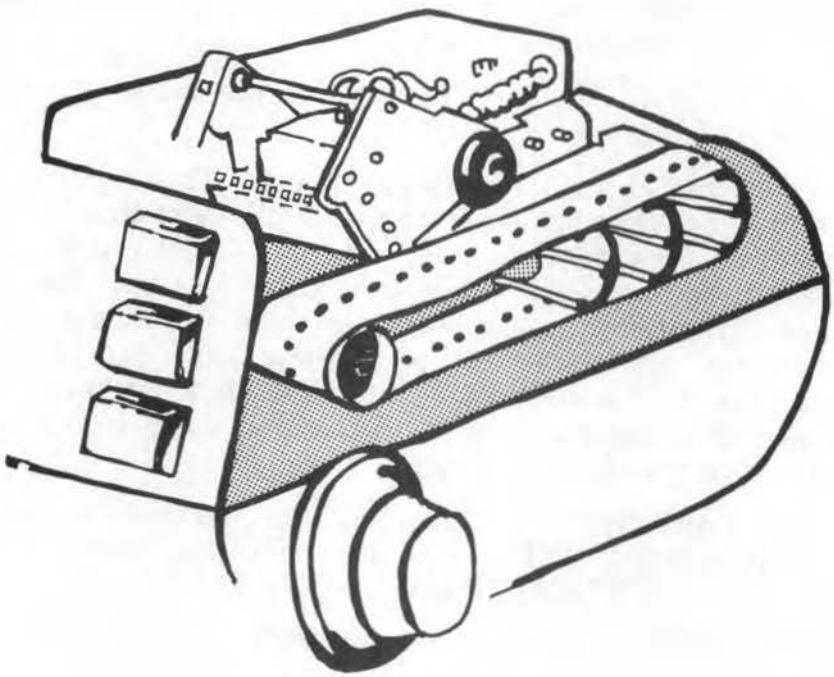


FIGURE 32. CARRIAGE CONTROL TAPE

The computer operator must set the tape and the first page of paper at line number one. After this adjustment, as a line is printed the carriage-control tape will advance a predetermined number of spaces, as will also the paper in the printer. Since the paper is the same size, it will advance from top to bottom as the tape makes one complete revolution. Making up carriage-control tapes is not a very difficult task after one or two tries and so we will not discuss it in detail. We should, however, take a further look at the printer spacing chart.

Fig. 33 shows the format of a report as it would appear on a spacing chart. Any printer spacing chart will serve two primary functions:

1. It establishes the positions of the data to be printed, and it indicates the spacing between lines. We must ask ourselves, for example, whether the printed report should show up on the left side, the middle, or the right side of a page.
2. It identifies each different type of line as either a heading, a detail, or a total.

31. We shall now discuss some of the more important features of Fig. 33.

The letter H appears opposite line 6. This reminds the programmer, when he is coding, that the first line on each page will be a heading line. The words PAYROLL and HOURS are printed on this line, using one box for each letter. In programming language these words are called "constants" or "literals." Use of these terms points to the fact that it is possible to create data in a program. All data to be processed does not necessarily come into a computer through an input device.

Line 7 also represents a heading line. On this line appear the abbreviation DEPT. and the spelled-out words NUMBER, NAME, WORKED, and NET PAY.

The letter D to the left of line 9 indicates that it is detail line. Each number or letter in the detail lines of a report is represented by an X.

Line 11 on the report represents a total line. There are eight positions reserved on this line for printing numbers. These positions also are represented by X's. The field for the total, shown under the heading NET PAY, contains a dollar sign, eight X's, a decimal point, and a comma. From

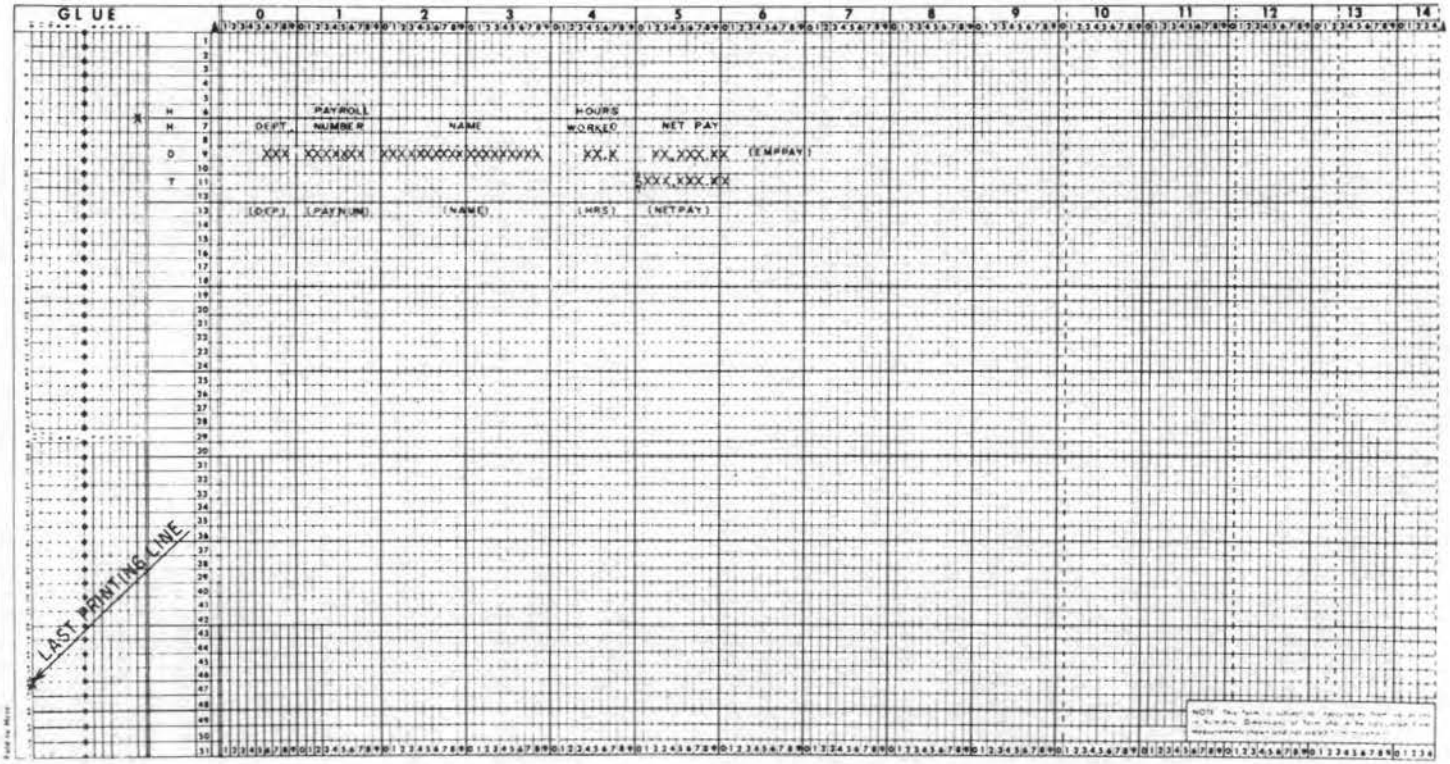


FIGURE 33. EXAMPLE OF A REPORT FORMAT ON A SPACING CHART

this example you can see that punctuation characters each use one block on the spacing chart, just as letters and numbers do.

For a programmer to be able to prepare a work sheet like the one in Fig. 33, he should have available either a copy of the desired report or a rough description of its form and size of each field. In addition, he must know what punctuation is to be included.

32. If a field is to be part of a line, you should place the letter X in each box where a character can print on the report. Each field to be printed should be assigned some field name so that the coder can refer to it easily. In Fig. 33 the field names are DEP, PAYNUM, NAME, HRS, NETPAY, and EMPPAY. Field names are usually enclosed within brackets to differentiate them from constants. An important rule about field names is that they should not be duplicated.

The rightmost position (the least significant one) in every field is called the field end position. For instance, at the top of the column for the field end position of NAME you see the number 8; and above that, and slightly to the left, is the number 3. The field end position for the field called (NAME) is 38, and the field end position for the field called (EMPPAY) is 60.

Before leaving the subject of printer spacing charts, there is one other thing to be said. If you must design a report for someone, ask that person to describe the report in detail. You may be able to suggest items he has forgotten. If possible, have him sketch out the form that the finished report should take.

Summary

33. Flow charts are frequently compared to road maps. Like road maps, they show the best route to follow and how to save time by taking shortcuts. A well-executed flow chart, however, is not a picture of the eventual coding; rather, it is a graphic representation of the logical steps in solving a problem. The flow chart helps to prevent coding errors in a proposed solution, and it aids the discovery of labor- and time-saving shortcuts. It also makes the basic elements in the method of solution much more understandable to anyone who has to participate in the work or to review it. A very important secondary benefit of flow charting is that it helps a

person to organize his thinking without recourse to some complex procedure. It could even be argued that this is the most important benefit of flow charting.

Because of the extreme time pressures which confront most data processing installations, a determined effort must be made to resist the temptation to save time by eliminating some of the flow charting. The elimination of needed flow charting makes for a decidedly shortsighted approach in terms of the overall man-hours required to complete a project. Even when actual mistakes are not made, there is sometimes the temptation to slight, to the detriment of the project, the drawing of the final version of the flow chart.

It goes without saying that flow charts must be kept up to date when revisions are made in a system. It is surprising how rapidly a flow chart can become obsolete, especially in the weeks and months following the introduction of a new system. It is at such a time that changes are most frequent.

The degree of flow-charting standardization which should be adopted varies with the needs of each installation. Where some people prepare the detailed flow charts and others do the coding, it is obvious that fairly rigid conventions must be established. When the same person does both the flow charting and the coding, standardization is still very desirable, but perhaps it need not be as rigid.

Check Your Learning

16. The standard form used for designing report formats is called the _____ _____ _____.

17. At the top of a carriage control tape there are twelve numbers, known as _____.

appendix

Answers to Check Your Learning

1.	condition, action	(Art. 2)
2.	four	(Art. 5)
3.	sequence	(Art. 8)
4.	flow of information	(Art. 10)
5.	sorting *or* collating	(Art. 12)
6.	processing, flow	(Art. 12)
7.	keying	(Art. 12)
8.	auxiliary	(Art. 12)
9.	last-record	(Art. 15)
10.	main line	(Art. 16)
11.	setup	(Art. 19)
12.	loop	(Art. 20)
13.	key-punch	(Art. 27)
14.	reference, classification, quantitative	(Art. 28)
15.	key	(Art. 29)
16.	printer spacing chart	(Art. 30)
17.	channels	(Art. 30)

fundamentals of computer programming

Serial 6396C-6 Part 3 Edition 1

Examination

Notice to Student. – Study this assignment thoroughly before you complete the following statements. Read each statement carefully and be sure you understand it. Each statement is followed by four words, phrases, or figures, only one of which is correct. Be sure to use the answer sheet provided. When you complete your work, check it carefully and correct any errors you may find. Then mail your completed answer sheet to us. DO NOT HOLD IT until another examination is ready.

1. Line numbers are shown vertically at the left-hand side of a printer spacing chart. What is indicated by the numbers that are shown horizontally across the top?
   A. Channel codes
   B. Heading lines
   C. Spacing and skipping
   D. Printing positions

2. Decision tables are related to the use of decision symbols in flow charting. To what type of coding instructions are they related?
   A. Input B. Output C. Editing D. Branching

3. The natural way to read a program flow chart is
   A. top to bottom and right to left.
   B. top to bottom and left to right.
   C. bottom to top and left to right.
   D. bottom to top and right to left.

4. A typical flow-chart form has
   A. 5 vertical rows, with 10 blocks in each row.
   B. 5 vertical rows, with 5 blocks in each row.
   C. 10 vertical rows, with 10 blocks in each row.
   D. 10 vertical rows, with 5 blocks in each row.

5. How many sections are contained in a standard decision table?
   A. Four B. Three C. Two D. One

6. On a flow chart, the fact that current output information will be used as input information in a subsequent processing cycle is shown by
   A. a communication link symbol.
   B. an on-page connector symbol.
   C. a broken line.
   D. a solid line.

7. The symbol used to indicate master or transaction records stored on storage devices is the
   A. document symbol.
   B. input/output symbol.
   C. auxiliary operation symbol.
   D. disk, drum, random-access file symbol.

8. Symbols on a system flow chart indicate the
   A. method of operation.
   B. details of each operation.
   C. order of operations.
   D. specific applications for operations.

9. With what type of programming instruction would the term "housekeeping" be most closely associated?
   A. Write B. Clear C. Add D. Branch

10. Arrowhead symbols should be used on flow charts to indicate
    A. an exit from a decision symbol.
    B. emphasis on a particular program instruction.
    C. that the flow of direction is not normal.
    D. that the chart is continued on a subsequent page.

11. A printer spacing chart is related to what type of program function?
    A. Input B. Output C. Processing D. Comparing

12. Instead of writing the same series of instructions for each card read, the programmer can use
    A. an exit.
    B. a loop.
    C. a decision table.
    D. a cross-reference.

13. Which type of line is usually related to the use of "constants" on the printer spacing chart?
    A. Grand total B. Detail C. Total D. Heading

14. Programmers should identify the possible different types of lines that a program may print out on a report. Three different types of lines on any printed report are
    A. heading, blank, and master.
    B. subtotal, total, and master.
    C. heading, detail, and total.
    D. detail, subtotal, and blank.

15. Channels on a carriage-control tape specify the
    A. area where glue should be applied.
    B. pages on which printing should begin and end.
    C. top and bottom margins on any page of a printout.
    D. twelve lines to be used for printing on each page of the printout.

16. "If you receive new items or cannot sell the old ones, request additional space." This instruction illustrates the OR problem statement because
    A. an alternative action will be needed.
    B. one action or the other will be executed.
    C. more than one condition must be met before an action takes place.
    D. there are two conditions, either of which can lead to a given action.

17. An invoice number punched in a card would be an example of what type of information?
    A. Classification B. Reference C. Quantitative D. Qualitative

18. The "field-end" position of any field on a printer spacing chart is the
    A. first blank character.
    B. last blank character.
    C. least significant character.
    D. most significant character.

19. In designing a card format for keypunching, what important requirement should be kept in mind?
    A. All alphabetic fields should be keypunched on the left-hand side of the cards.
    B. The number of special symbols should be kept to a minimum.
    C. Information should be arranged to correspond to the source document.
    D. Master information should be punched in one type of card and transaction data in another type.

20. A symbol on a flow chart should represent a
    A. program routine.
    B. program subroutine.
    C. program application.
    D. program instruction.

ICS

Scranton, Pennsylvania 18515

FOR YOUR INSTRUCTOR'S USE	
GRADE	GRADED BY

ANSWER SHEET

YOUR STUDENT NO.
–

Fundamentals of Computer Programming, Part 3
6396C-6
Edition 1

PLEASE PRINT

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

☐ Check if this is a new address and you have not previously notified us.

FOLD

INDICATE YOUR ANSWER TO EACH QUESTION BY MARKING AN X
IN THE APPROPRIATE SQUARE. EXAMPLE: [X] B C D

1.	A	B	C	D	11.	A	B	C	D
2.	A	B	C	D	12.	A	B	C	D
3.	A	B	C	D	13.	A	B	C	D
4.	A	B	C	D	14.	A	B	C	D
5.	A	B	C	D	15.	A	B	C	D
6.	A	B	C	D	16.	A	B	C	D
7.	A	B	C	D	17.	A	B	C	D
8.	A	B	C	D	18.	A	B	C	D
9.	A	B	C	D	19.	A	B	C	D
10.	A	B	C	D	20.	A	B	C	D

CUT ALONG THIS LINE

FOLD

☐ INDICATE THE TOTAL NUMBER OF HOURS YOU SPENT STUDYING THIS LESSON AND COMPLETING THIS EXAMINATION.